WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Wednesday, March 31, 2004

Hide?	Set Name	Query	Hit Count
		DB=EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L18	L17	0
		DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L17	L16 and (instruction near loop$)	79
☐	L16	L8 and (single near instruction)	92
		DB=EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L15	L14	0
		DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L14	L13 and single near instruction	49
☐	L13	L8 and l3	102
		DB=DWPI; PLUR=YES; OP=ADJ	
☐	L12	L8	530
☐	L11	L9	0
☐	L10	L9	0
		DB=EPAB; PLUR=YES; OP=ADJ	
☐	L9	L8	63
		DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L8	L7 or l6 or l5	4065
☐	L7	l6 or multiloop or multilooping or multi?looping	1021
☐	L6	multi?loop	811
☐	L5	(multiple near loop) or (multiple near looping) or multiple?loop or multiple?looping	3189
☐	L4	(multiple near loop$) or multiple?loop or multiple?looping	3356
		DB=PGPB,USPT,USOC; PLUR=YES; OP=ADJ	
☐	L3	L2 or l1	7684
☐	L2	712/10-24,204-213,220-221,226-227,230-248.ccls.	6258
☐	L1	717/114,119,131-133,149-161.ccls.	1598

END OF SEARCH HISTORY

WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Wednesday, March 31, 2004

Hide?	Set Name	Query	Hit Count
		DB=PGPB,USPT,USOC; PLUR=YES; OP=ADJ	
☐	L14	20020083305	1
		DB=USPT,PGPB,JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L13	L12 and (single near instruction) same loop	1
☐	L12	multi?loop	762
		DB=DWPI; PLUR=YES; OP=ADJ	
☐	L11	L6	1
		DB=EPAB; PLUR=YES; OP=ADJ	
☐	L10	L6	0
		DB=TDBD; PLUR=YES; OP=ADJ	
☐	L9	L6	0
		DB=USPT,PGPB,JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L8	L7 and multiple branch	4
		DB=USPT; PLUR=YES; OP=ADJ	
☐	L7	L6	77
		DB=USPT,PGPB,JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L6	L5 and fetch$	96
☐	L5	L4 and (loop near instruction)	125
☐	L4	pipeline$ near loop	366
		DB=PGPB,USPT,USOC; PLUR=YES; OP=ADJ	
☐	L3	L2 or l1	4984
☐	L2	712/10-24,204-213.ccls.	3465
☐	L1	717/114,119,131-133,149-161.ccls.	1598

END OF SEARCH HISTORY

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 20 of 49 returned.

☐ 1. Document ID: US 20040024998 A1

Using default format because multiple data bases are involved.

L14: Entry 1 of 49 File: PGPB Feb 5, 2004

PGPUB-DOCUMENT-NUMBER: 20040024998
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20040024998 A1

TITLE: System to dispatch several instructions on available hardware resources

PUBLICATION-DATE: February 5, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Chauvel, Gerald	Antibes		FR	

US-CL-CURRENT: 712/241

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Ir

☐ 2. Document ID: US 20030204840 A1

L14: Entry 2 of 49 File: PGPB Oct 30, 2003

PGPUB-DOCUMENT-NUMBER: 20030204840
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20030204840 A1

TITLE: Apparatus and method for one-pass profiling to concurrently generate a frequency profile and a stride profile to enable data prefetching in irregular programs

PUBLICATION-DATE: October 30, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Wu, Youfeng	Palo Alto	CA	US	

US-CL-CURRENT: 717/158; 717/160

ABSTRACT:

An apparatus and method for one-pass profiling to concurrently generate a frequency

profile and a stride profile to enable pre-fetching of irregular program data are described. In one embodiment, the method includes the selective generation of stride profile information according to partially generated frequency profile information to concurrently form a stride profile and a frequency profile during execution of a user program instrumented during a single profiling pass. Once the stride profile and frequency profile are generated, prefetch instructions are inserted into the user program utilizing the stride profile and the frequency profile. In one embodiment, the present invention utilizes profiling to identify regular stride patterns in irregular program code, which is referred to herein as stride profiling. Consequently, by identifying regular stride patterns within the irregular program code, one embodiment of the invention enables prefetching of irregular program data to reduce system stalls due to data cache misses.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | In

---

☐ 3. Document ID: US 20030177343 A1

L14: Entry 3 of 49 File: PGPB Sep 18, 2003

PGPUB-DOCUMENT-NUMBER: 20030177343
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20030177343 A1

TITLE: Methods and apparatus for multi-processing execution of computer instructions

PUBLICATION-DATE: September 18, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Magoshi, Hidetaka	Palo Alto	CA	US	

US-CL-CURRENT: 712/241; 712/235

ABSTRACT:

A multi-processing computer architecture and a method of operating the same are provided. The multi-processing architecture provides a main processor and multiple sub-processors cascaded together to efficiently execute loop operations. The main processor executes operations outside of a loop and controls the loop. The multiple sub-processors are operably interconnected, and are each assigned by the main processor to a given loop iteration. Each sub-processor is operable to receive one or more sub-instructions sequentially, operate on each sub-instruction and propagate the sub-instruction to a subsequent sub-processor.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | In

---

☐ 4. Document ID: US 20020083305 A1

L14: Entry 4 of 49 File: PGPB Jun 27, 2002

PGPUB-DOCUMENT-NUMBER: 20020083305

PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20020083305 A1

TITLE: Single instruction for multiple loops

PUBLICATION-DATE: June 27, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Renard, Pascal L.	Annemasse	TX	FR	
Gergen, Joseph P.	Manchaca		US	

US-CL-CURRENT: 712/220

ABSTRACT:

Embodiments of the present invention relate generally to the manner in which processors execute multiple loop instructions. That is, embodiments of the invention relate to the organization of multiple loop constructs, such as, for example, nested loops, to achieve improved performance during loop execution. One embodiment contemplates a single instruction that provides for execution of other instructions of a set of instructions in accordance with multiple looping constructs. Another embodiment contemplates a single-loop instruction suitable for terminating on multiple termination conditions.

☐ 5. Document ID: US 6615340 B1

L14: Entry 5 of 49 File: USPT Sep 2, 2003

US-PAT-NO: 6615340
DOCUMENT-IDENTIFIER: US 6615340 B1

TITLE: Extended operand management indicator structure and method

DATE-ISSUED: September 2, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Wilmot, II; Richard Byron	Lafayette	CA	94549	

US-CL-CURRENT: 712/209; 712/217, 712/218

ABSTRACT:

Extended operand management indicators stored during initial program execution enable management and regulation of operand values and streamline their handling. Operand values are stored in new types of stores. Operand location management indicators indicate current operand value locations among various store types for selected operands. Indicated operand-forwarding policies for selected operands streamline forwarding of operand values from source instructions to value receiving target instructions. Indicated loop iterations of operand source instructions enable forwarding

of operands over more than one loop iteration. Stride indicators indicate strides of program loop accesses to matrix operands. Inter-loop indicators enable forwarding of operand values from source loop instructions directly to target loop instructions. Constant or nearly constant operands are indicated to enable their storage in special caches. Operands used for cross-CPU serialization are indicated for special handling and storage in spin lock cache. Indicators of farthest back and farthest forward branches since operand last update are used to enhance the prediction of longer-range branch directions. Virtual predicate operand indicators streamline execution of densely branching program code. Stack operand indicators enable nullification of paired stack pointer increment-decrement operations to avoid serious operand serialization bottlenecks in very high issue rate machines.

15 Claims, 19 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 19

☐ 6. Document ID: US 6321330 B1

L14: Entry 6 of 49 File: USPT Nov 20, 2001

US-PAT-NO: 6321330
DOCUMENT-IDENTIFIER: US 6321330 B1
**** See image for Certificate of Correction ****

TITLE: Each iteration array selective loop data prefetch in multiple data width prefetch system using rotating register and parameterization to avoid redundant prefetch

DATE-ISSUED: November 20, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Doshi; Gautam B.	Sunnyvale	CA		
Muthukumar; Kalyan	Cupertino	CA		

US-CL-CURRENT: 712/241; 711/213, 712/225

ABSTRACT:

The present invention provides a mechanism for prefetching array data efficiently from within a loop. A prefetch instruction is parameterized by a register from a set of rotating registers. On each loop iteration, a prefetch is implemented according to the parameterized prefetch instruction, and the address targeted by the prefetch instruction is adjusted. The registers are rotated for each loop iteration, and the prefetch instruction parameterized by the rotating register is adjusted accordingly. The number of iterations between prefetches for a given array is determined by the number of elements in the set of rotating register.

35 Claims, 3 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 3

☐ 7. Document ID: US 6286135 B1

L14: Entry 7 of 49 File: USPT Sep 4, 2001

US-PAT-NO: 6286135
DOCUMENT-IDENTIFIER: US 6286135 B1

TITLE: Cost-sensitive SSA-based strength reduction algorithm for a machine with predication support and segmented addresses

DATE-ISSUED: September 4, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Santhanam; Vatsa	Campbell	CA		

US-CL-CURRENT: 717/146; 717/150, 717/151

ABSTRACT:

A compiler optimization algorithm that deals with aggressive strength reduction of integer machine instructions found in loops. The algorithm permits the strength reduction of such machine instructions whose execution may be guarded by predicate values. In addition, the algorithm allows the strength reduction of address calculations consumed by memory reference instructions accessing data in a segmented virtual address space. The algorithm also permits aggressive SSA-based strength reduction of non-address integer computations found in loops that are linear functions of loop induction variables. The algorithm incorporates profitability considerations by reducing the cost of updating strength-reduction temporaries and ensures that the strength-reduction transformation results in an overall reduction of the path-lengths within loop bodies, without creating excessive register pressure.

55 Claims, 72 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 27

☐ 8. Document ID: US 6178499 B1

L14: Entry 8 of 49 File: USPT Jan 23, 2001

US-PAT-NO: 6178499
DOCUMENT-IDENTIFIER: US 6178499 B1

TITLE: Interruptable multiple execution unit processing during operations utilizing multiple assignment of registers

DATE-ISSUED: January 23, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Stotzer; Eric	Houston	TX		
Scales; Richard H.	Houston	TX		

US-CL-CURRENT: 712/241; 712/244

ABSTRACT:

A method of operating a multiple execution unit microprocessor in a software pipelined loop is disclosed. This method allows the microprocessor to respond to interrupt requests and other runtime conditions during execution of a software pipelined loop utilizing multiple assignment of registers. In one embodiment, the method comprises branching out of the software pipelined loop, upon occurrence of an interrupt request, to an interrupt epilog that consumes in-flight register values and sets the interrupt return pointer to the address of an interrupt prolog. The interrupt is then taken. The interrupt prolog is a subset of the loop prolog, and restores the processor to an operational state equivalent to one that would have existed had the interrupt not been taken. Loop execution is then resumed without data loss or corruption.

16 Claims, 4 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 2

☐ 9. Document ID: US 6173394 B1

L14: Entry 9 of 49 File: USPT Jan 9, 2001

US-PAT-NO: 6173394
DOCUMENT-IDENTIFIER: US 6173394 B1

TITLE: Instruction having bit field designating status bits protected from modification corresponding to arithmetic logic unit result

DATE-ISSUED: January 9, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Guttag; Karl M.	Missouri City	TX		
Poland; Sydney W.	Katy	TX		
Balmer; Keith	Bedford			GB

US-CL-CURRENT: 712/226; 708/525, 712/221, 712/224

ABSTRACT:

A data processing apparatus includes plural data registers, an arithmetic logic unit and a status register. The status register stores a plurality of different types of status bits. These status bits could be a negative status bit, a carry status bit, an overflow status bit and a zero status bit. These status bits are normally set dependent upon the

condition of the result generated by the current arithmetic logic unit operation. A status bit protect instruction type permits selection of status bits protected from modification corresponding to the current arithmetic logic unit result. This status bit protect instruction preferably includes individual protect bit corresponding to each status bit. If a protect bit has a first digital state, then the corresponding status bit may be modified corresponding to the current arithmetic logic unit result. If the protect bit has a second opposite digital state, then the corresponding status bit is protected from modification according to the arithmetic logic unit results.

19 Claims, 71 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 37

☐ 10. Document ID: US 6139199 A

L14: Entry 10 of 49 File: USPT Oct 31, 2000

US-PAT-NO: 6139199
DOCUMENT-IDENTIFIER: US 6139199 A

TITLE: Fast just-in-time (JIT) scheduler

DATE-ISSUED: October 31, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Rodriguez; John E.	Santa Cruz	CA		

US-CL-CURRENT: 717/159; 712/214, 712/217, 712/23, 717/118, 717/148, 717/155

ABSTRACT:

A just-in-time (JIT) compiler typically generates code from bytecodes that have a sequence of assembly instructions forming a "template". It has been discovered that a just-in-time (JIT) compiler generates a small number, approximately 2.3, assembly instructions per bytecode. It has also been discovered that, within a template, the assembly instructions are almost always dependent on the next assembly instruction. The absence of a dependence between instructions of different templates is exploited to increase the size of issue groups using scheduling. A fast method for scheduling program instructions is useful in just-in-time (JIT) compilers. Scheduling of instructions is generally useful for just-in-time (JIT) compilers that are targeted to in-order superscalar processors because the code generated by the JIT compilers is often sequential in nature. The disclosed fast scheduling method has a complexity, and therefore an execution time, that is proportional to the number of instructions in an instruction block (N complexity), a substantial improvement in comparison to the N.sup.2 complexity of conventional compiler schedulers. The described fast scheduler advantageously reorders instructions with a single pass, or few passes, through a basic instruction block while a conventional compiler scheduler such as the DAG scheduler must iterate over an instruction basic block many times. A fast scheduler operates using an analysis of a sliding window of three instructions, applying two rules within the three instruction window to determine when to reorder instructions. The analysis includes acquiring the opcodes and operands of each instruction in the three instruction window,

and determining register usage and definition of the operands of each instruction with respect to the other instructions within the window. The rules are applied to determine ordering of the instructions within the window.

36 Claims, 7 Drawing figures
Exemplary Claim Number: 14
Number of Drawing Sheets: 7

☐ 11. Document ID: US 6125440 A

L14: Entry 11 of 49 File: USPT Sep 26, 2000

US-PAT-NO: 6125440
DOCUMENT-IDENTIFIER: US 6125440 A

TITLE: Storing executing instruction sequence for re-execution upon backward branch to reduce power consuming memory fetch

DATE-ISSUED: September 26, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Osovets; Alexander	Ashburn	VA		

US-CL-CURRENT: 712/205; 711/109, 712/241

ABSTRACT:

A controller for a digital processor includes a random access memory, e.g., an instruction memory, that consumes significant power when operating. To reduce the power consumption when repetitive instructions, i.e. loops, are being performed, the instructions being executed are stored in a shift register and, when a jump-back instruction is executed, the instructions, including those in the loop, are then accessed from the shift register rather than from the random access memory without any additional special instructions that define the characteristics of the loop. A memory control includes a state tracking machine that monitors the execution of the program instructions and determines from the execution of a jump-back instruction that a loop may have been entered, whereupon it enables the shift register to produce the instructions stored therein and disables the instruction memory from producing instructions that are stored in the shift register. The foregoing process preferably is automatically initiated for each loop, whether the loop is a new loop, a loop within a loop or a multiple loop.

50 Claims, 4 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 4

☐ 12. Document ID: US 6116768 A

L14: Entry 12 of 49 File: USPT Sep 12, 2000

US-PAT-NO: 6116768
DOCUMENT-IDENTIFIER: US 6116768 A

TITLE: Three input arithmetic logic unit with barrel rotator

DATE-ISSUED: September 12, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Guttag; Karl M.	Missouri City	TX		
Balmer; Keith	Bedford			GB
Gove; Robert J.	Plano	TX		
Read; Christopher J.	Houston	TX		
Golston; Jeremiah E.	Sugar Land	TX		
Poland; Sydney W.	Kary	TX		
Ing-Simmons; Nicholas	Huntingdon			GB
Moyse; Phillip	Bromham			GB

US-CL-CURRENT: 708/236; 708/209, 712/221

ABSTRACT:

A data processing apparatus includes a three input arithmetic logic unit (230) that generates a combination of the three inputs that is selected by a function signal. Data registers (200) store the three data inputs and the arithmetic logic unit output. The second input signal comes from a controllable barrel rotator (235). The rotate amount is a default rotate amount stored in a special data register, a predetermined set of bits of data recalled from a data register or zero. A one's constant source (236) is connected to the barrel rotator (235) to supply a multibit digital signal of "1". This permits generating a second input signal of the form 2.sup.N, with N being the rotate amount. The output of the barrel rotator (235) may be stored independently of the arithmetic logic unit (230) result. In the preferred embodiment of this invention, the three input arithmetic logic unit (230) is embodied in a data processor circuits as a part of a multiprocessor integrated circuit (100) used in image processing.

25 Claims, 70 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 37

☐ 13. Document ID: US 6098163 A

L14: Entry 13 of 49 File: USPT Aug 1, 2000

US-PAT-NO: 6098163
DOCUMENT-IDENTIFIER: US 6098163 A

TITLE: Three input arithmetic logic unit with shifter

DATE-ISSUED: August 1, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Guttag; Karl M.	Missouri City	TX		
Balmer; Keith	Bedford			GB
Gove; Robert J.	Plano	TX		
Read; Christopher J.	Houston	TX		
Golston; Jeremiah E.	Sugar Land	TX		
Poland; Sydney W.	Katy	TX		
Ing-Simmons; Nicholas	Huntingdon			GB
Moyse; Phillip	Bedford			GB

US-CL-CURRENT: 712/20

ABSTRACT:

A data processing apparatus includes a three input arithmetic logic unit (230) that generates a combination of the three inputs that is selected by a function signal. Data registers (200) store the three data inputs and the arithmetic logic unit output. The second input signal comes from a controllable shifter (235). The shift amount is a default shift amount stored in a special data register, a predetermined set of bits of data recalled from a data register or zero. A one's constant source (236) is connected to the shifter (235) to supply a multibit digital signal of "1". This permits generating a second input signal of the form 2.sup.N, with N being the shift amount. The output of the shifter (235) may be stored independently of the arithmetic logic unit (230) result. The third input signal comes from a multiplexer (233) that selects between an instruction specified immediate field, data recalled from a data register or a mask input from a mask generator (239). One preferred form of the mask has a number of right justified 1's corresponding to a mask input signal. This mask input signal may be the default shift amount or a predetermined number of the least significant bits of a third input signal as selected by a multiplexer. A second preferred form of the mask is selected one of the left most 1, the right most 1, the left most bit change or the right most bit change of a predetermined set of the least significant bits of data recalled from a data register. In the preferred embodiment of this invention, the three input arithmetic logic unit (230) is embodied in a data processor circuits as a part of a multiprocessor integrated circuit (100) used in image processing.

31 Claims, 70 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 37

☐ 14. Document ID: US 6085275 A

L14: Entry 14 of 49 File: USPT Jul 4, 2000

US-PAT-NO: 6085275
DOCUMENT-IDENTIFIER: US 6085275 A
**** See image for Certificate of Correction ****

TITLE: Data processing system and method thereof

DATE-ISSUED: July 4, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Gallup; Michael G.	Austin	TX		
Goke; L. Rodney	Austin	TX		
Seaton, Jr.; Robert W.	Austin	TX		
Lawell; Terry G.	Austin	TX		
Osborn; Stephen G.	Austin	TX		
Tomazin; Thomas J.	Austin	TX		

US-CL-CURRENT: 710/316; 370/355, 712/10

ABSTRACT:

A data processing system (55) and method thereof includes one or more data processors (10). Data processor (10) is capable of performing both vector operations and scalar operations. Using a single microsequencer (22), data processor (10) is capable of executing both vector instructions and scalar instructions. Data processor (10) also has a memory circuit (14) capable of storing both vector operands and scalar operands.

47 Claims, 319 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 196

☐ 15. Document ID: US 6079008 A

L14: Entry 15 of 49 File: USPT Jun 20, 2000

US-PAT-NO: 6079008
DOCUMENT-IDENTIFIER: US 6079008 A

TITLE: Multiple thread multiple data predictive coded parallel processing system and method

DATE-ISSUED: June 20, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Clery, III; William B.	Rockville	MD		

US-CL-CURRENT: 712/11; 712/13, 712/16, 712/20, 712/43

ABSTRACT:

A parallel processing system or processor has a computing architecture including a plurality of execution units to repeatedly distribute instruction streams within the

processor via corresponding buses, and a series of processing units to access the buses and selectively execute the distributed instruction streams. The execution units each retrieve an instruction stream from an associated memory and place the instruction stream on a corresponding bus, while the processing units individually may select and execute any instruction stream placed on the corresponding buses. The processing units autonomously execute conditional instructions (e.g., IF/ENDIF instructions, conditional looping instructions, etc.), whereby an enable flag within the processing unit is utilized to indicate occurrence of conditions specified within a conditional instruction and control selective execution of instructions in response to occurrence of those conditions. An enable stack is utilized to facilitate processing and execution of nested conditional instructions by storing the states of the enable flag for each nested conditional instruction. The parallel processor may further delay placement of selected instruction streams onto corresponding buses until each processing unit selecting a particular instruction stream enters a state to execute that instruction stream. In

addition, each execution unit may cease placing an instruction stream onto a corresponding bus in response to no processing units selecting that instruction stream for execution.

12 Claims, 7 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 6

☐ 16. Document ID: US 6058473 A

L14: Entry 16 of 49 File: USPT May 2, 2000

US-PAT-NO: 6058473
DOCUMENT-IDENTIFIER: US 6058473 A

TITLE: Memory store from a register pair conditional upon a selected status bit

DATE-ISSUED: May 2, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Guttag; Karl M.	Missouri City	TX		
Poland; Sydney W.	Katy	TX		
Balmer; Keith	Bedford			GB

US-CL-CURRENT: 712/225; 712/226, 712/234

ABSTRACT:

A memory store operation comes from one of a pair of registers selected by an arithmetic logic unit condition. An instruction logic circuit (250, 660) controls an addressing circuit (120) to store data in a first register into memory if a selected status bit has a first state and to store data in a second register associated with the first register into memory if the selected status bit has a second state in response to a register pair conditional store instruction. The bits may indicate a negative output of the arithmetic logic unit (230), a carry out signal, an overflow, or a zero output. The register pair conditional store instruction designates a particular one of the status bits to control

the

conditional store. The instruction logic circuit (250, 660) substitutes the selected status bit for a least significant bit of the register number. Thus memory store is from the first register if the status bit is "1" and is from the second register if the status bit is "0". In a further embodiment the register pair conditional write instruction is conditional. The write operation aborts if the designated condition is true. In the preferred embodiment of this invention, the arithmetic logic unit (230), the status register (210), the data registers (200) and the instruction decode logic (250, 660) are embodied in at least one digital image/graphics processor (71) as a part of a multiprocessor formed in a single integrated circuit (100) used in image processing.

57 Claims, 70 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 37

☐ 17. Document ID: US 6055627 A

L14: Entry 17 of 49 File: USPT Apr 25, 2000

US-PAT-NO: 6055627
DOCUMENT-IDENTIFIER: US 6055627 A

TITLE: Compiling method of accessing a multi-dimensional array and system therefor

DATE-ISSUED: April 25, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Kyushima; Ichiro	Yokohama			JP
Kainaga; Masahiro	Yokohama			JP

US-CL-CURRENT: 712/233; 712/234

ABSTRACT:

Particular portions of a program is determined to execute repeatedly. Array references accessed in a loop are classified to groups of equal values on the basis of reference to the same array. Of the groups, the ones for which an array transposition can use a batch load or store instruction are selected. An array corresponding to the selected groups is transposed to generate an intermediate language for copying. Reference to the elements of the array before the transposition is changed to reference to the transposition array. This makes it possible to convert the loop so that a single instruction can be used for a processor which can load into a register two data having consecutive addresses in a memory with use of the instruction.

8 Claims, 19 Drawing figures
Exemplary Claim Number: 8
Number of Drawing Sheets: 15

☐ 18. Document ID: US 6044222 A

L14: Entry 18 of 49 File: USPT Mar 28, 2000

US-PAT-NO: 6044222
DOCUMENT-IDENTIFIER: US 6044222 A

TITLE: System, method, and program product for loop instruction scheduling hardware lookahead

DATE-ISSUED: March 28, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Simons; Barbara Bluestein	Palo Alto	CA		
Sarkar; Vivek	Newton	MA		

US-CL-CURRENT: 717/156; 712/207, 712/233, 712/237, 712/238, 712/241, 717/161

ABSTRACT:

Improved scheduling of instructions within a loop for execution by a computer system having hardware lookahead is provided. A dependence graph is constructed which contains all the nodes of a dependence graph corresponding to the loop, but which only contains loop-independent dependence edges. A start node simulating a previous iteration of the loop may be added to the dependence graph, and an end node simulating a next iteration of the loop may also added to the dependence graph. A loop-independent edge between a source node and the start node is added to the dependence graph, and a loop-independent edge between a sink node and the end node is added to the dependence graph. Loop-carried edges which satisfy a computed lower bound on the time required for a single loop iteration are eliminated from a dependence graph, and loop-carried edges which do not satisfy the computed lower bound are replaced by a pair of loop-independent edges. Instructions may be scheduled for execution based on the dependence graph.

21 Claims, 13 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 9

☐ 19. Document ID: US 6026484 A

L14: Entry 19 of 49 File: USPT Feb 15, 2000

US-PAT-NO: 6026484
DOCUMENT-IDENTIFIER: US 6026484 A

TITLE: Data processing apparatus, system and method for if, then, else operation using write priority

DATE-ISSUED: February 15, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Golston; Jeremiah E.	Sugar Land	TX		

US-CL-CURRENT: 712/226; 712/225, 712/234

ABSTRACT:

A data processing apparatus employs write priority to permit a data processing apparatus to execute an if, then, else operation in a single instruction cycle. The data processing apparatus includes pipelined data unit (110) and address unit (120) operations. The address unit (120) data move operation has a higher write priority than the storing of the data unit (110) operation. The data unit (110) includes an arithmetic logic unit (230) that performs an unconditional operation with the result to be stored in a destination register (200). The address unit (120) sets the address for a data move operation to the same destination register (200). The data move operation is conditional upon the if condition set by the instruction and based upon a set of status bits in a status register (210). The status register (210) includes a plurality of status bits set corresponding to a prior arithmetic logic unit (230) result. The status bits preferably include a negative status bit, a carry status bit, an overflow status bit and a zero status bit. This address unit (120) data move operation, having a higher write priority than the data unit (110) operation, controls the data written into the destination register (200). If the status bits do not match the condition specified in the instruction, then the conditional data move does not take place and the results of the data unit operation are stored in the destination register (200).

96 Claims, 69 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 37

☐ 20. Document ID: US 6003128 A

L14: Entry 20 of 49 File: USPT Dec 14, 1999

US-PAT-NO: 6003128
DOCUMENT-IDENTIFIER: US 6003128 A

TITLE: Number of pipeline stages and loop length related counter differential based end-loop prediction

DATE-ISSUED: December 14, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Tran; Thang M.	Austin	TX		

US-CL-CURRENT: 712/241; 712/23, 712/239

ABSTRACT:

An apparatus for prediction of loop instructions. Loop instructions decrement the value in a counter register and branch to a target address (specified by an instruction operand) if the decremented value of the counter register is greater than zero. The apparatus comprises a loop detection unit that detects the presence of a loop instruction in the instruction stream. An indication of the loop instruction is conveyed to a reorder buffer which stores speculative register values. If the apparatus is not currently processing the loop instruction, a compare value corresponding to the counter register prior to execution of the loop instruction is conveyed to a loop prediction unit. The loop prediction unit also increments a counter value upon receiving each indication of the loop instruction. This counter value is then compared to the compare value conveyed from the reorder buffer. If the counter value is one less than the compare value, a signal is asserted that indicates that the loop instruction should be predicted not-taken upon a next iteration of the loop. In this manner, loop prediction accuracy may be increased by correctly predicting the loop instruction not-taken. Because loops are commonly found in a variety of applications, increasing the accuracy of loop prediction, even slightly, may have a beneficial effect on performance. The loop operation is particularly important in scientific applications where it may be used to perform various digital signal processing routines and to traverse arrays.

19 Claims, 12 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 12

Terms	Documents
L13 and single near instruction	49